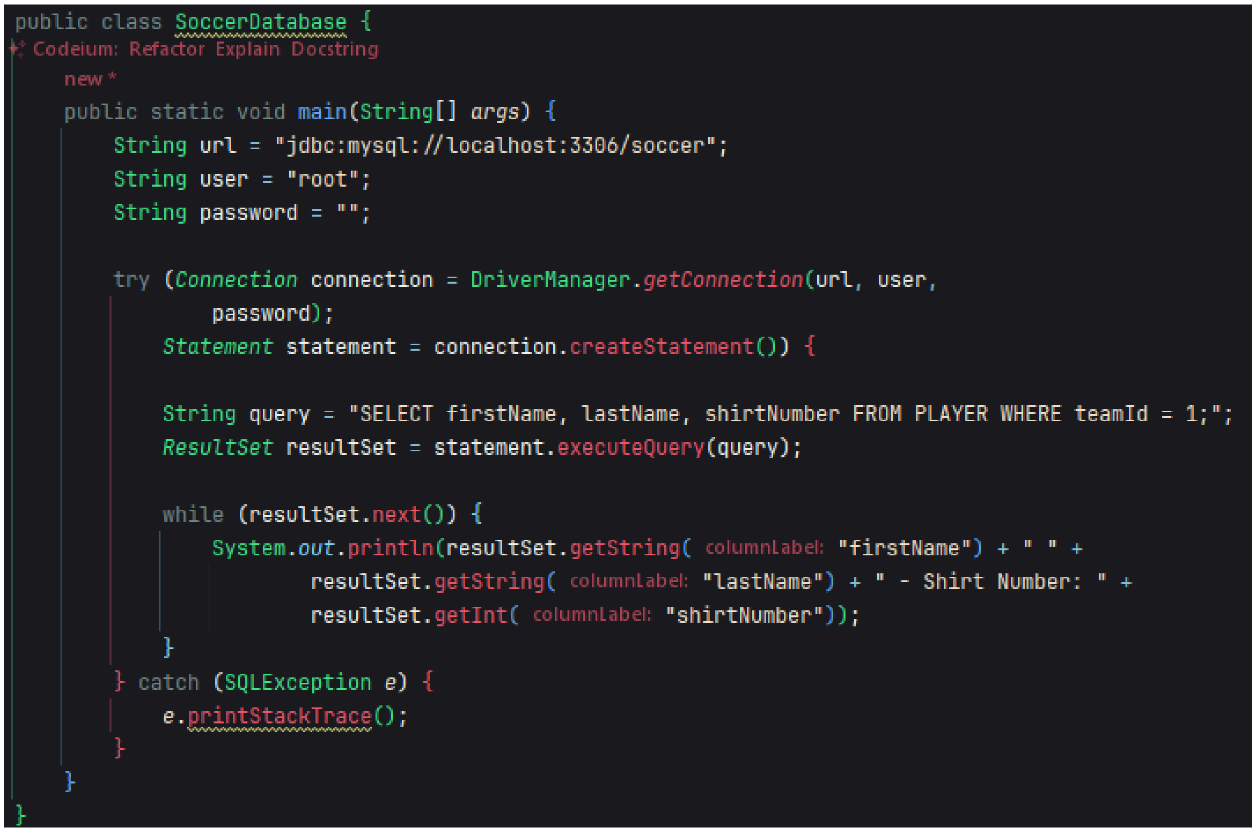
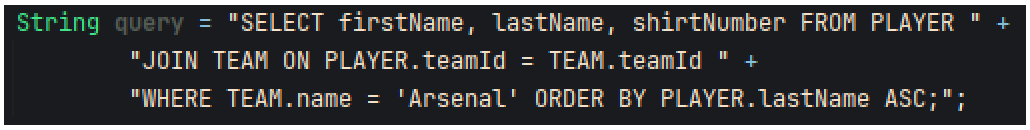
1. Get a list of players names and shirt numbers that play on the team with id = 1 import java.sql.*;



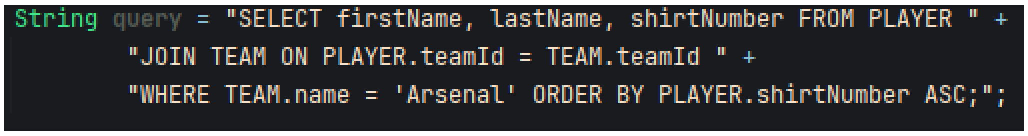
```
public class SoccerDatabase {
✦ Codeium: Refactor Explain Docstring
    new *
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/soccer";
        String user = "root";
        String password = "";

        try (Connection connection = DriverManager.getConnection(url, user,
                password);
            Statement statement = connection.createStatement()) {

            String query = "SELECT firstName, lastName, shirtNumber FROM PLAYER WHERE teamId = 1;";
            ResultSet resultSet = statement.executeQuery(query);

            while (resultSet.next()) {
                System.out.println(resultSet.getString( columnLabel: "firstName") + " " +
                        resultSet.getString( columnLabel: "lastName") + " - Shirt Number: " +
                        resultSet.getInt( columnLabel: "shirtNumber"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

2. Get a list of players names and shirt numbers that play on the team named “Arsenal" and display them in alphabetic order based on players last name.



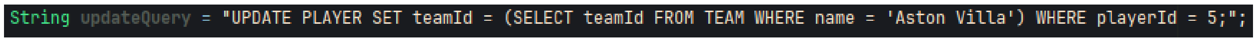
```
String query = "SELECT firstName, lastName, shirtNumber FROM PLAYER " +
         "JOIN TEAM ON PLAYER.teamId = TEAM.teamId " +
         "WHERE TEAM.name = 'Arsenal' ORDER BY PLAYER.lastName ASC;";
```

3. Get a list of players names and shirt numbers that play on the team named “Arsenal" and display them in numeric order based on players shirt number



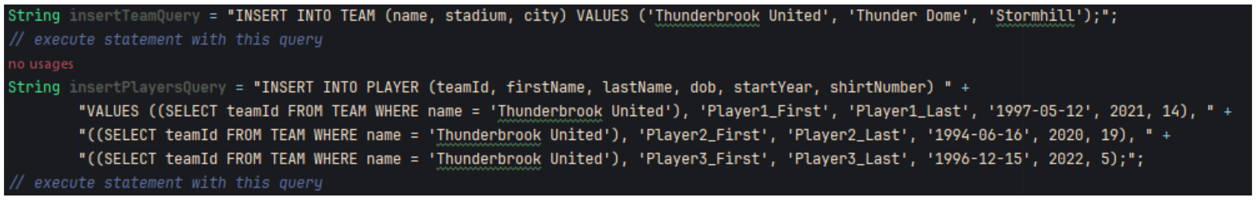
```
String query = "SELECT firstName, lastName, shirtNumber FROM PLAYER " +
         "JOIN TEAM ON PLAYER.teamId = TEAM.teamId " +
         "WHERE TEAM.name = 'Arsenal' ORDER BY PLAYER.shirtNumber ASC;";
```

4. Update a players team to be any team you would like. ex. update player with player id =5 to be on team “Aston Villa"



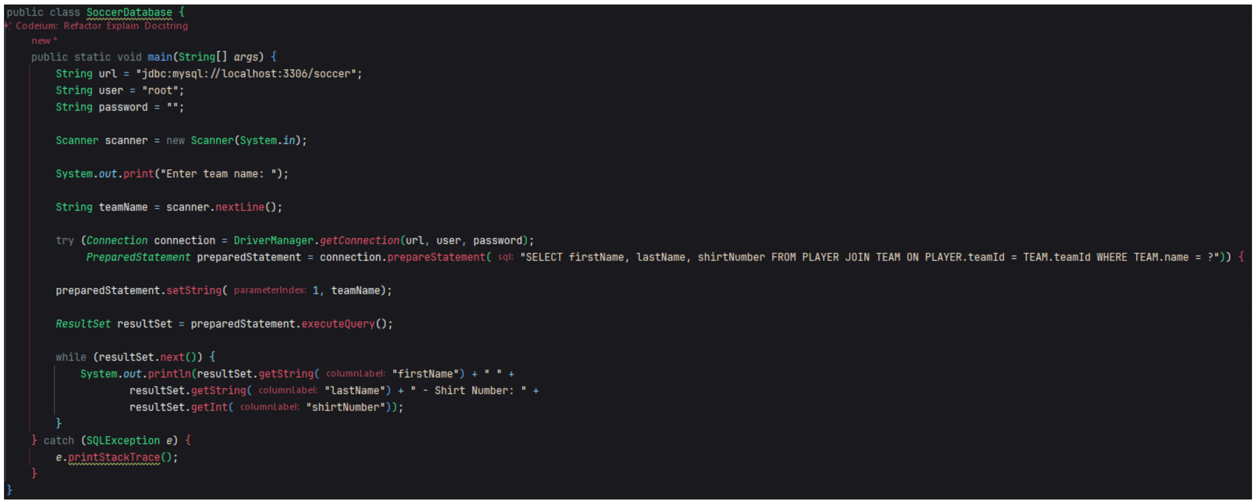
```
String updateQuery = "UPDATE PLAYER SET teamId = (SELECT teamId FROM TEAM WHERE name = 'Aston Villa') WHERE playerId = 5;";
```

5. Add a new team to the database. You may research a real team or make up your own. Form statements to add 3 new players to the new team.



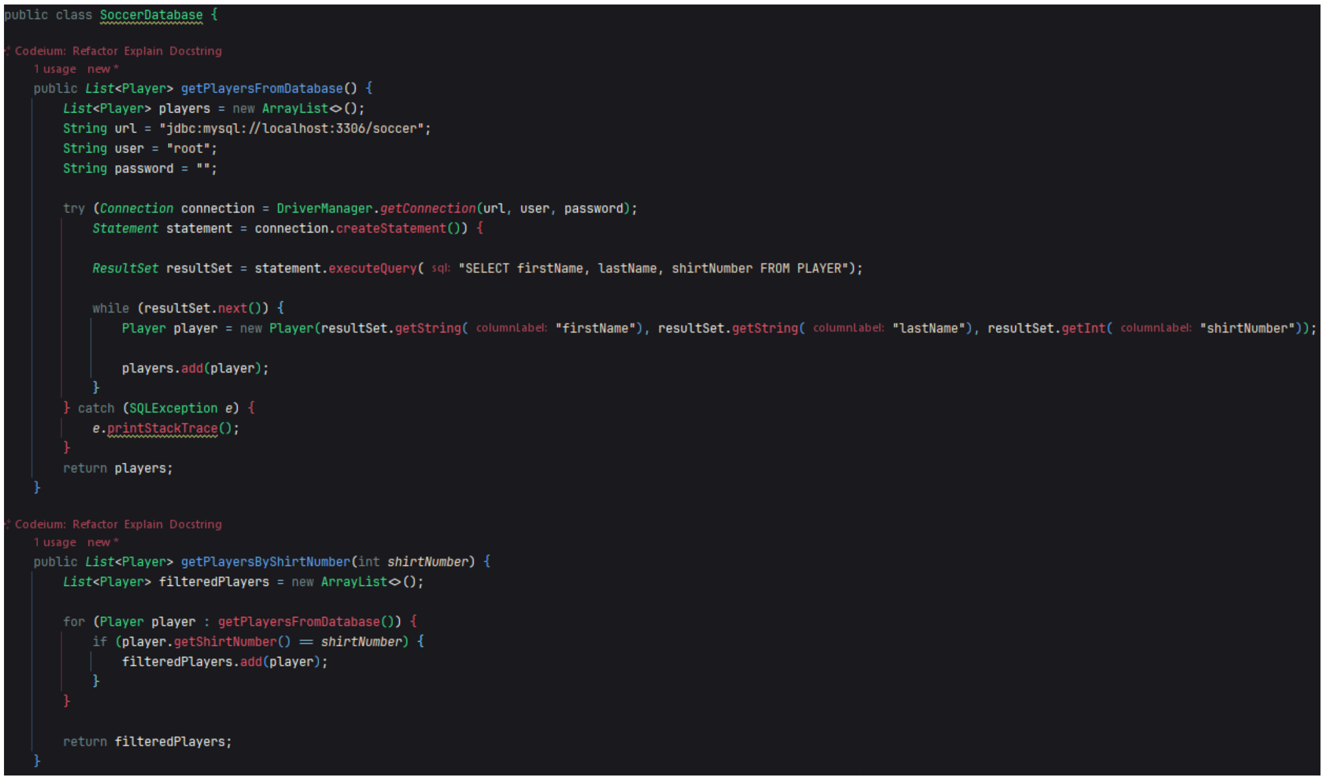
```
String insertTeamQuery = "INSERT INTO TEAM (name, stadium, city) VALUES ('Thunderbrook United', 'Thunder Dome', 'Stormhill');";
// execute statement with this query
no usages
String insertPlayersQuery = "INSERT INTO PLAYER (teamId, firstName, lastName, dob, startYear, shirtNumber) " +
        "VALUES ((SELECT teamId FROM TEAM WHERE name = 'Thunderbrook United'), 'Player1_First', 'Player1_Last', '1997-05-12', 2021, 14), " +
        "((SELECT teamId FROM TEAM WHERE name = 'Thunderbrook United'), 'Player2_First', 'Player2_Last', '1994-06-16', 2020, 19), " +
        "((SELECT teamId FROM TEAM WHERE name = 'Thunderbrook United'), 'Player3_First', 'Player3_Last', '1996-12-15', 2022, 5);";
// execute statement with this query
```

6. Write a program that asks the user using keyboard or command line arguments to enter the name of a team that they want the players for.

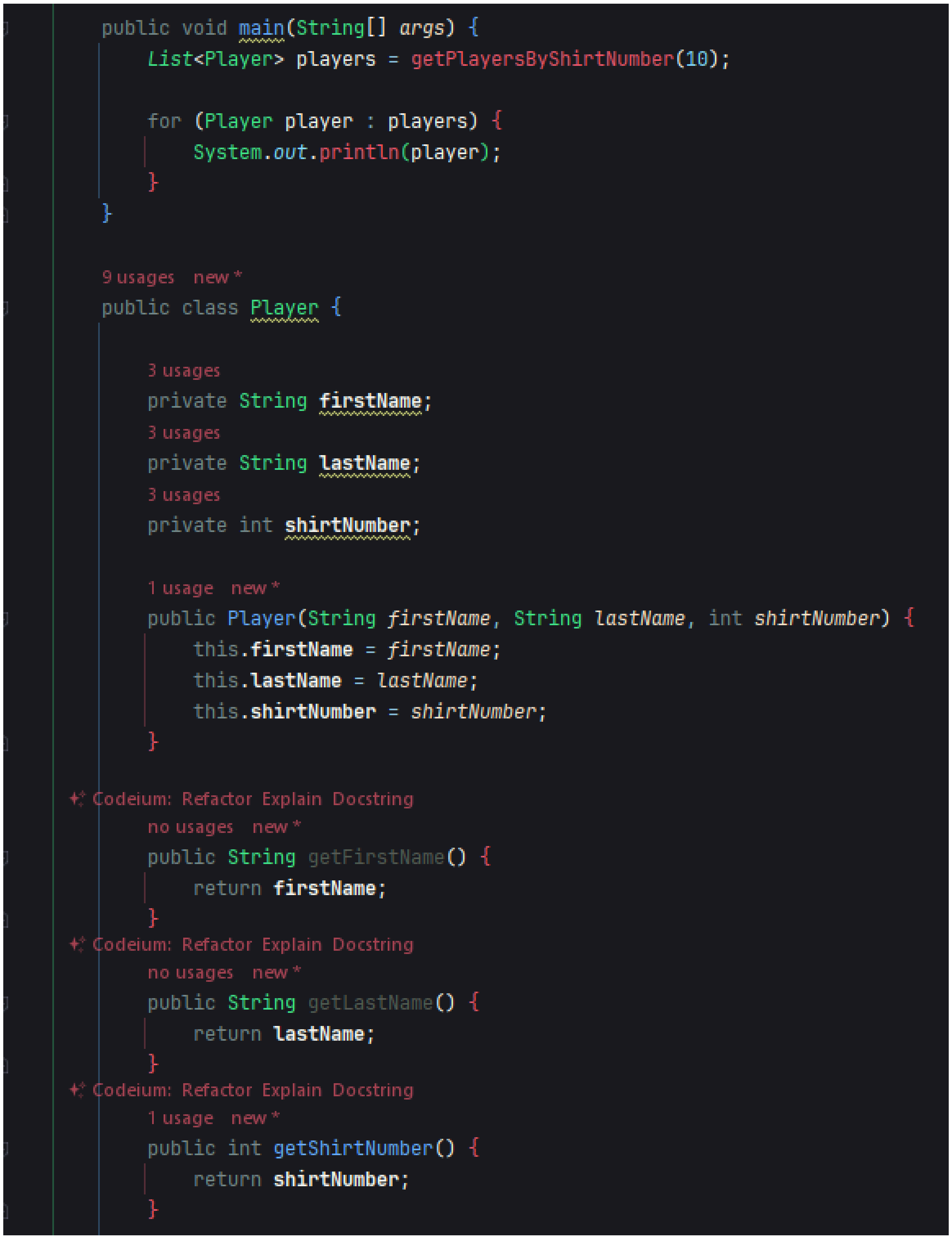


```
public class SoccerDatabase {
 Codeium: Refactor Explain Docstring
    new *
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/soccer";
        String user = "root";
        String password = "";

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter team name: ");

        String teamName = scanner.nextLine();

        try (Connection connection = DriverManager.getConnection(url, user, password);
             PreparedStatement preparedStatement = connection.prepareStatement( sql: "SELECT firstName, lastName, shirtNumber FROM PLAYER JOIN TEAM ON PLAYER.teamId = TEAM.teamId WHERE TEAM.name = ?")) {

        preparedStatement.setString( parameterIndex: 1, teamName);

        ResultSet resultSet = preparedStatement.executeQuery();

        while (resultSet.next()) {
            System.out.println(resultSet.getString( columnLabel: "firstName") + " " +
                    resultSet.getString( columnLabel: "lastName") + " - Shirt Number: " +
                    resultSet.getInt( columnLabel: "shirtNumber"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

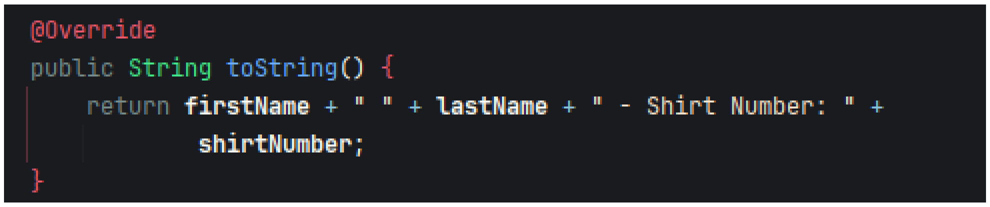
7. Write a Java class that captures the nature of a player. Write a method to create a list or array of player objects where each object has been populated with data from the database. Write a method to return all the players that have shirt number x, where x is a parameter.



```java
public class SoccerDatabase {

 Codeium: Refactor Explain Docstring
    1 usage  new *
    public List<Player> getPlayersFromDatabase() {
        List<Player> players = new ArrayList<>();
        String url = "jdbc:mysql://localhost:3306/soccer";
        String user = "root";
        String password = "";

        try (Connection connection = DriverManager.getConnection(url, user, password);
            Statement statement = connection.createStatement()) {

            ResultSet resultSet = statement.executeQuery( sql: "SELECT firstName, lastName, shirtNumber FROM PLAYER");

            while (resultSet.next()) {
                Player player = new Player(resultSet.getString( columnLabel: "firstName"), resultSet.getString( columnLabel: "lastName"), resultSet.getInt( columnLabel: "shirtNumber"));

                players.add(player);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return players;
    }

 Codeium: Refactor Explain Docstring
    1 usage  new *
    public List<Player> getPlayersByShirtNumber(int shirtNumber) {
        List<Player> filteredPlayers = new ArrayList<>();

        for (Player player : getPlayersFromDatabase()) {
            if (player.getShirtNumber() == shirtNumber) {
                filteredPlayers.add(player);
            }
        }

        return filteredPlayers;
    }
```



```
    public void main(String[] args) {
        List<Player> players = getPlayersByShirtNumber(10);

        for (Player player : players) {
            System.out.println(player);
        }
    }

    9 usages   new *
    public class Player {

        3 usages
        private String firstName;
        3 usages
        private String lastName;
        3 usages
        private int shirtNumber;

        1 usage   new *
        public Player(String firstName, String lastName, int shirtNumber) {
            this.firstName = firstName;
            this.lastName = lastName;
            this.shirtNumber = shirtNumber;
        }

  Codeium: Refactor  Explain  Docstring
        no usages   new *
        public String getFirstName() {
            return firstName;
        }
  Codeium: Refactor  Explain  Docstring
        no usages   new *
        public String getLastName() {
            return lastName;
        }
  Codeium: Refactor  Explain  Docstring
        1 usage   new *
        public int getShirtNumber() {
            return shirtNumber;
        }
```



```java
@Override
public String toString() {
    return firstName + " " + lastName + " - Shirt Number: " +
            shirtNumber;
}
```